UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/385,394	08/30/1999	JOHN S. YATES JR.	114596/03-4000.	9093

7590 02/11/2004

Intellectual Property Legal Assistants
WILLKIE, FARR & GALLAGHER, LLP
787 Seventh Ave.
New York, NY 10019

EXAMINER	
ELLIS, RICHARD L	

ART UNIT	PAPER NUMBER
2183	38

DATE MAILED: 02/11/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/385,394	YATES ET AL.	
	Examiner	Art Unit	
	Richard Ellis	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE _____ MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *03 July 2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-133* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-19,21-33,37-47,49-59,61-85,87-126 and 128-133* is/are rejected.

7)☒ Claim(s) *20,34-36,48,60,86 and 127* is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

1. Claims 1-133 are presented for examination.

2. The computer program listing filed on [1] as a "microfiche appendix" is unacceptable. A computer program listing conforming to the requirements of 37 CFR 1.96 is required.

The microfiche appendix is unacceptable because it is not limited to containing only a computer program listing. There appear to be at least six tables, and two graphical flowcharts contained within the frames of the appendix. Both of which are not a computer program listing as defined by MPEP 608.05(a). Applicant is reminded that in resubmitting the appendix in proper format, he must adhere to the new 37 CFR 1.96 rules regarding submission of a computer program listing because as of February 28, 2001, the office no longer accepts microfiche.

3. The attempt to incorporate subject matter into this application by reference to US Patent applications 09/322,443, 09/298,536, and 09/239,194 is improper because those applications themselves incorporate by reference additional subject matter. See MPEP 608.01(p)(A).

4. The attempt to incorporate subject matter into this application by reference to a list of publications on pg. 143 is improper because applicant may not incorporate essential material by reference to a document which is not a US patent or US patent publication.

5. Claims 61-84 are rejected under 35 USC § 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

5.1. The following terms lack proper antecedent basis:

5.1.1. "the virtual memory manager" claim 61;

6. The following is a quotation of the appropriate paragraphs of 35 USC § 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless --
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

7. The following is a quotation of the appropriate paragraphs of 35 USC § 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless --

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a)shall have the effects for the purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

8. The following is a quotation of 35 USC § 103 which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

(c) Subject matter developed by another person, which qualifies as prior art only under one or more of subsections (e), (f), and (g) of section 102 of this title, shall not preclude patentability under this section where the subject matter and the claimed invention were, at the time the invention was made, owned by the same person or subject to an obligation of assignment to the same person.

9. This application currently names joint inventors. In considering patentability of the claims under 35 USC § 103, the examiner presumes that the subject matter of the various claims was commonly owned at the time any inventions covered therein were made absent any evidence to the contrary. Applicant is advised of the obligation under 37 CFR § 1.56 to point out the inventor and invention dates of each claim that was not commonly owned at the time a later invention was made in order for the examiner to consider the applicability of potential 35 USC § 102(f) or (g) prior art under 35 USC § 103.

10. Claims 1-2, 4-14, 17, 37-47, 51-53, 61-66, 68, 70-72, 104-109, 113-115, and 128-132 are rejected under 35 USC § 102(e) as being clearly anticipated by Goetz et al., U.S. Patent 5,854,913.

Goetz et al. taught (e.g. see figs. 1-16) the invention as claimed (as per claim 1), including a data processing ("DP") system comprising:

10.1. A computer (fig. 1), comprising:

10.2. a processor pipeline (10) designed to alternately execute instructions coded for first and second different computer architectures or coded to implement first and second different processing conventions (col. 1 lines 11-17);

10.3. a memory (col. 7 lines 22-29) for storing instructions for execution by the processor pipeline (col. 8 lines 9-13), the memory being divided into pages (fig. 4, fig. 5, col. 10 lines 4-42) for management by a virtual memory manager (col. 5 lines 1-7), a single address space of the memory having first and second pages (col. 17 lines 13-16 and 24-28);

10.4. a memory unit (fig. 1, 102) designed to fetch instructions from the memory for execution by the pipeline, and to fetch stored indicator elements (fig. 10 and col. 15

lines 6-10) associated with respective memory pages of the single address space from which the instructions are to be fetched (col. 14 lines 18-22), each indicator element designed to store an indication of which of two different computer architectures and/or execution conventions under which instruction data of the associated page are to be executed by the processor pipeline (col. 17 lines 24-33), the indicator elements being architecturally addressable when the processor pipeline is executing under one of the architectures or data storage conventions (col. 18 lines 59-65), and architecturally unaddressable when the processor pipeline is executing under the other architecture or data storage convention (col. 12 lines 30-43 and col. 18 lines 59-65);

10.5. the memory unit and/or processor pipeline further designed to recognize an execution flow from the first page, whose associated indicator element indicates the first architecture or execution convention, to the second page, whose associated indicator element indicates the first architecture or execution convention (col. 14 line 50 to col. 15 line 19), and in response to the recognizing, to adapt a processing mode of the processor pipeline or a storage content of the memory to effect execution of instructions in the architecture and/or under the convention indicated by the indicator element corresponding to the instruction's page (col. 15 lines 11-19).

11. As to claim 2, Goetz et al. taught that the two architectures are two instruction set architectures (col. 1 lines 11-16); and wherein the adapting includes controlling instruction execution hardware of the computer to interpret the instructions according to the two instruction set architectures according to the indicator elements (col. 4 lines 55-60 and col. 6 lines 1-10).

12. As to claim 4, Goetz et al. taught:

executing instructions fetched from first and second regions of a memory of a computer (col. 7 lines 22-29 and col. 17 lines 13-16 and 24-28), the instructions of the first and second regions being coded for execution by computers of first and second architectures or following first and

second data storage conventions, respectively (col. 1 lines 11-17), the memory regions having associated first and second indicator elements (fig. 10 and col. 14 lines 18-22), the indicator elements each having a value indicating the architecture or data storage convention under which instructions, from the associated region are to be executed (col. 17 lines 24-33), the indicator elements being architecturally addressable when the processor pipeline is executing under one of the architectures or data storage conventions (col. 18 lines 59-65), and architecturally unaddressable when the computer is executing under the other architecture or data storage convention (col. 12 lines 30-43 and col. 18 lines 59-65); when execution of the instruction data flows or transfers from the first region to the second, adapting the computer for execution in the second architecture or convention (col. 15 lines 11-19).

13. As to claim 5, Goetz et al. taught that the regions were pages (fig. 4, fig. 5, col. 10 lines 4-42) managed by a virtual memory manager (col. 5 lines 1-7).

14. As to claim 6, Goetz et al. taught that the indicator elements were stored in a table of indicator elements (col. 14 lines 19-22, 54-62) distinct from a primary address translation table used by the virtual memory manager (col. 17 lines 13-16), the indicator elements of the table associated with corresponding pages of the memory (inherent, the definition of page table entry means that the entry is associated with a corresponding page of memory).

15. As to claim 7, Goetz et al. taught that the indicator elements are stored in a table (see above), each indicator element associated with a corresponding physical page frame (col. 3 line 64 to col. 4 line 20).

16. As to claim 8, Goetz et al. taught that the entries are entries of a translation look-aside buffer (col. 17 lines 13-16).

17. As to claim 9, Goetz et al. taught that the regions were lines of an instruction cache (fig. 1, 104, col. 17 lines 28-32).

18. As to claim 10, Goetz et al. taught that the two architectures are two instruction set architectures (col. 1 lines 11-17); and wherein the adapting step includes controlling instruction execution hardware of the computer to interpret the instructions according to the two

instruction set architectures according to the indicator elements (col. 15 lines 59-67).

19. As to claim 11, Goetz et al. taught that the regions are pages (fig. 4, fig. 5, col. 10 lines 4-42) managed by a virtual memory manager (col. 5 lines 1-7).

20. As to claim 12, Goetz et al. taught that the indicator elements are stored in a table (col. 10 lines 4-42) whose entries are indexed by physical page frame number (fig. 4, 50, col. 10 lines 20-22).

21. As to claim 13, Goetz et al. taught that the entry is one entry of a translation look-aside buffer (col. 17 lines 13-16).

22. As to claim 14, Goetz et al. taught that a mode of execution of the instructions is changed without software intervention when execution flows or transfers from the first region to the second (col. 17 lines 22-49).

23. As to claim 17, Goetz et al. taught that one of the regions stores an off-the-shelf operating system binary (col. 4 lines 41-45) coded in an instruction set non-native to the computer (col. 19 lines 5-8), the non-native instruction set providing access to a reduced subset of the resources of the computer (col. 12 lines 30-43).

24. As to claim 37, Goetz et al. taught:

24.1. a processor pipeline (10) configured to alternately execute instructions of computers of two different architectures or processing conventions (col. 1 lines 11-17); and

24.2. a memory unit (fig. 1, 102) designed to fetch instructions from a computer memory for execution by the pipeline, and

24.3. to fetch stored indicator elements (fig. 10 and col. 15 lines 6-10) associated with respective memory regions of a single address space from which the instructions are to be fetched (col. 14 lines 18-22),

24.4. each indicator element designed to store an indication of the architecture or execution convention under which the instruction data of the associated region are to be executed by the processor pipeline (col. 17 lines 24-33),

24.5. the indicator elements being maintained in storage that is architecturally addressable when the processor pipeline is executing in one of the computer architectures or processing conventions (col. 18 lines 59-65), and architecturally unaddressable when the processor pipeline is executing in the other architecture or convention (col. 12 lines 30-43 and col. 18 lines 59-65);

24.6. the memory unit and/or processor pipeline further designed to recognize an execution flow or transfer from a region whose indicator element indicates one architecture or execution convention to another (col. 14 line 50 to col. 15 line 19).

25. As to claim 38, Goetz et al. taught the indicator elements are stored in a table of indicator elements (col. 14 lines 19-22, 54-62) distinct from a primary address translation table used by the virtual memory manager (col. 17 lines 13-16), the indicator elements of the table associated with corresponding pages of the memory (inherent, the definition of page table entry means that the entry is associated with a corresponding page of memory).

26. As to claim 39, Goetz et al. taught a translation look-aside buffer (TLB) (col. 17 lines 13-16); and TLB control circuitry designed to load the indicator elements into the TLB from a table stored in memory (col. 14 line 63 to col. 15 line 10 and col. 17 lines 13-16), the entries of the table being indexed by associated with corresponding physical page frame number (col. 3 line 64 to col. 4 line 20).

27. As to claim 40, Goetz et al. taught that the two architectures are two instruction set architectures (col. 1 lines 11-17), and further that the processor pipeline control circuitry was designed to control the processor pipeline to effect interpretation of the instructions under the two instruction set architectures alternately, according to the associated indicator elements (col. 4 lines 55-60 and col. 6 lines 1-10).

28. As to claim 41, Goetz et al. taught software programmed to manage a transition between the execution of a program executing in the first instruction set architecture, being an instruction set architecture native to the computer processor (col. 17 lines 61-64), and execution of an off-the-shelf operating system (col. 4 lines 41-45) coded in the second

instruction set, being an instruction set non-native to the computer (col. 19 lines 5-8), providing access to a reduced subset of the resources of the computer (col. 12 lines 30-43).

29. As to claim 51, Goetz et al. taught:

29.1. storing instructions in pages (fig. 4, fig. 5, col. 10 lines 4-42) of a computer memory managed by a virtual memory manager (col. 5 lines 1-7),

29.2. the instruction data of the pages being coded for execution by, respectively, computers of two different architectures and/or under two different execution conventions (col. 1 lines 11-17);

29.3. in association with pages of the memory, storing corresponding indicator elements indicating the architecture or convention in which the instructions of the pages are to be executed (col. 17 lines 24-33);

29.4. executing instructions from the pages in a common processor (10, col. 6 lines 1-25),

29.5. the processor designed, responsive to the page indicator elements, to execute instructions in the architecture or under the convention indicated by the indicator element corresponding to the instruction's page (col. 17 lines 24-33).

30. As to claim 52, Goetz et al. taught that the pages' indicator elements are stored in a table whose entries are indexed by physical page frame number (fig. 4, 50, col. 10 lines 20-22), and cached in a translation look-aside buffer (col. 17 lines 13-16).

31. As to claim 53, Goetz et al. taught that the two architectures are two instruction set architectures (col. 1 lines 11-16), and further taught the step of controlling the instruction execution hardware of the computer to interpret the instructions according to the two instruction set architectures according to the indicator element (col. 4 lines 55-60 and col. 6 lines 1-10).

32. As to claim 61, Goetz et al. taught:

32.1. an instruction unit (fig. 1, 105), configured to fetch instructions from a memory (col. 7 lines 22-29) managed by the virtual memory manager (col. 5 lines 1-7), and

32.2. configured to execute instructions coded for first and second different computer architectures or coded to implement first and second different data storage conventions (col. 1 lines 11-17);

32.3. the microprocessor chip being designed (a) to retrieve indicator elements stored in association with respective pages of the memory (fig. 10 and col. 15 lines 6-10), each indicator element indicating the architecture or convention in which the instructions of the page are to be executed (col. 17 lines 24-33), and

32.4. (b) to recognize when instruction execution has flowed or transferred from a page of the first architecture or convention to a page of the second, as indicted by the respective associated indicator elements (col. 17 lines 34-49), and

32.5. (c) to alter a processing mode of the instruction unit or a storage content of the memory to effect execution of instructions in accord with the indicator element associated with the page of the second architecture or convention (col. 17 lines 34-49).

33. As to claim 62, Goetz et al. taught that the indicator elements are stored in virtual address translation table entries (col. 17 lines 28-33).

34. As to claim 63, Goetz et al. taught that the indicator elements are stored in a table (col. 14 lines 19-22, 54-62) distinct from a primary address translation table used by a virtual memory manager (col. 17 lines 13-16), the indicator elements of the table being stored in association with respective pages of the memory (inherent, the definition of page table entry means that the entry is associated with a corresponding page of memory).

35. As to claim 64, Goetz et al. taught that the indicator elements are stored in association with respective physical page frames (col. 3 line 64 to col. 4 line 20).

36. As to claim 65, Goetz et al. taught that the indicator elements are stored in association with respective virtual pages (col. 17 lines 28-34).

37. As to claim 66, Goetz et al. taught that the indicator elements are stored in entries of a translation look-aside buffer (col. 17 lines 13-16).

38. As to claim 68, Goetz et al. taught that a mode of execution of the instructions is changed without software intervention when execution flows or transfers from the first region to the second (col. 17 lines 22-49).

39. As to claim 70, Goetz et al. taught that the architecture or convention indicator elements were stored in a table whose entries are indexed by physical page frame number (fig. 4, 50, col. 10 lines 20-22), and cached in a translation look-aside buffer (col. 17 lines 13-16).

40. As to claim 71, Goetz et al. taught that the two architectures were two instruction set architectures (col. 1 lines 11-16), and the microprocessor chip controlled the instruction unit to interpret the instructions according to the two instruction set architectures according to the indicator element corresponding to the pages from which the instructions are fetched (col. 4 lines 55-60 and col. 6 lines 1-10).

41. As to claim 72, Goetz et al. taught software programmed to manage a transition between the execute8on of a program executing in the first instruction set architecture, being an instruction set architecture native to the computer processor (col. 17 lines 61-64), and execution of an off-the-shelf operating system (col. 4 lines 41-45) coded in the second instruction set, being an instruction set non-native to the computer (col. 19 lines 5-8) providing access to a

42. As to claim 104-109, 113-115, 128-132, they do not teach or define above the invention claimed in the previously respective rejected claims and are therefore rejected under Goetz et al. for the same reasons set fourth in the previous claim rejections, supra.

43. Claims 94-95 are rejected under 35 USC § 102(b) as being clearly anticipated by Brender et al., U.S. Patent 5,339,422.

44. As to claim 94, Brender et al. taught executing a section of computer object code twice (Murphy et al. fig. 7, fig. 8), without modification of the code section between the two executions (Brender et al. col. 10 lines 15-40), the code section materializing a destination address into a register and being architecturally defined to directly transfer control indirectly through the register to the destination address (Brender et al. at col. 21 lines 41-49 and col. 22

lines 1-14), the two executions materializing two different destination addresses (col. 21 lines 41-44); the two destination code sections at the two materialized destination addresses being, coded in two distinct instruction sets and, respectively, obeying the default calling conventions native to each of the two instruction sets, neither instruction set being a subset of the others (col. 21 lines 47-49).

45. As to claim 95, Brender et al. taught that the code at the first destination receives floating-point arguments and returns floating-point return values using a register-based calling convention (Murphy et al. at fig. 7, 262, fig. 8, 306), and;

the code at the second destination receives floating-point arguments using a memory-based stack calling convention, and returns floating-point values using a register indicated by a top-of-stack pointer (Murphy et al. at fig. 7, 268, 270, 272, fig. 8, 292, 294).

46. Claims 3, 15-16, 18-19, 21-33, 42-47, 49-50, 54-59, 69, 73-85, 87-93, 96-103, 110-112, 116-126, and 133 are rejected under 35 USC § 103 as being unpatentable over Goetz et al., U.S. Patent 5,854,913, in view of Brender et al., U.S. Patent 5,339,422 and Murphy et al., U.S. Patent 5,764,947 (incorporated by reference into Brender et al. at col. 1 lines 10-12 and 19-24).

47. As to claims 3 and 18, Goetz et al. did not teach that the two conventions were first and second calling conventions and in response to detection of flow from a first region to a second, to alter the data storage content of the computer to create a program context under the second calling convention that is logically equivalent to a pre-alteration program context under the first calling convention. However, Brender et al. taught a multiple architecture system (col. 4 lines 1-22, referred to as "domains" in Brender et al.) which provided for cross domain (inter-architecture) calls (col. 4 lines 29-46) having different calling conventions (col. 2 line 61 to col. 3 line 2 and col. 9 line 35 to col. 10 line 40). Brender et al. taught the necessity of transforming the "data storage content" of the computer in order to create "a program context under the second calling convention that is logically equivalent to a pre-alteration program context under the first calling convention (Brender et al., fig. 5, col. 16 lines 11-45, Murphy et

al. figs. 7-8, col. 8 line 56 to col. 10 lines 57). It would have been obvious to a person of ordinary skill in the art at the time the invention was made to have combined the teachings of Brender et al. with Goetz et al. because Brender et al. specifically states that conversion of calling conventions must be performed when a cross domain function call is performed (see Murphy et al. at col. 3 lines 1-22). Brender et al. additionally details why it is advantageous to provide cross domain calls at col. 5 lines 12-55.

48. As to claim 15, Goetz et al. did not teach that the computer takes an exception when execution flows or transfers from the first region to the second. However, Brender et al. taught use of an exception to signal transfer from a first region to a second (col. 8 lines 57-67). It would have been obvious to a person of ordinary skill in the art at the time the invention was made to have used generation of an exception to signal code flow in Goetz et al.'s system because of Brender et al.'s teaching that use of an exception was a preferred method of such signaling (col. 8 lines 66-67).

49. As to claim 16, Brender et al. taught that the mode of execution of the instructions is explicitly controlled by an exception handler (col. 8 line 68 to col. 9 line 4, Brender et al.'s "jacketing routine" is responsive to the hardware trap, and is therefore the exception handler).

50. As to claim 19, Goetz et al. taught that the regions were pages (fig. 4, fig. 5, col. 10 lines 4-42) managed by a virtual memory manager (col. 5 lines 1-7). Goetz et al. did not teach the claimed data storage conventions. However, Brender et al. taught that one of the two data storage conventions was a register-based calling convention (col. 9 lines 36-40, and 67), and the other data storage convention is a memory stack-based calling convention (col. 9 line 68).

51. As to claim 21, Brender et al. taught that the instruction data coded for execution by a first of the two instruction set architectures observes a data storage convention associated with the first architecture (col. 2 lines 61-65 and col. 10 lines 50-64), and instruction data coded for execution by a second of the two instruction set architectures observes a second, different, data storage convention associated with the second architecture (col. 10 lines 50-64), a single indicator element indicating both the instruction set architecture and the data storage

convention (col. 8 lines 60-61 and 66-67, Goetz et al. at col. 17 lines 28-31);

and further comprising the step of recognizing when program execution flows or transfers from a region using the first instruction set architecture to a region using the second instruction set architecture (col. 8 lines 57-63), and in response to the recognition, adjusting the data storage content of the computer from the first storage convention to the second (fig. 5, Murphy et al. in figs. 7-8).

52. As to claim 22, Goetz et al. in view of Brender et al. and Murphy et al. taught:

52.1. executing instructions fetched from first and second regions of a memory of a computer (Goetz et al. at col. 7 lines 22-29 and col. 17 lines 13-16 and 24-28),

52.2. the instructions of the first and second regions being coded for execution by computers following first and second data storage conventions (Goetz et al. at col. 1 lines 11-17, Brender et al. at col. 9 line 35 to col. 10 line 40),

52.3. the memory regions having associated first and second indicator elements (Goetz et al. at fig. 10 and col. 14 lines 18-22),

52.4. the indicator elements each having a value indicating the data storage convention under which instructions from the associated region are to be executed (Goetz et al. col. 17 lines 24-33);

52.5. recognizing when program execution has flowed or transferred from a region whose indicator element indicates the first data storage convention to a region whose indictor element indicates the second data storage convention (Goetz et al. at col. 17 lines 34-49),

52.6. and in response to the recognition, altering the data storage content of the computer to create a program context under the second data storage convention that is logically equivalent to a pre-alteration program context under the first data storage convention (Brender et al. at fig. 5, col. 16 lines 11-45, Murphy et al. at figs. 7-8, col. 8 line 56 to col. 10 lines 57).

53. It would have been obvious to combine the teachings of Brender et al. and Murphy et al. with that of Goetz et al. for the same reasons presented, supra. in the rejection of earlier claims under Goetz et al. in view of Brender et al. and Murphy et al.

54. As to claim 23, Goetz et al. taught overlaying the logical resources of the first and second instruction set architectures onto the physical resources of the computer according to a mapping that assigns corresponding resources of the two architectures to a common physical resource of a computer when the resources serve analogous functions in the calling conventions of the two architectures (col. 6 lines 1-25).

55. As to claim 24, Brender et al. and Murphy et al. taught that the adjusting step further comprised altering a bit representation of a datum from a first representation in the first convention to a second representation in the second convention, the alteration of representation being chosen to preserve the meaning of the datum across the change in data storage convention (Brender et al. in fig. 5, element 316, Murphy et al. in figs. 7-8).

56. As to claim 25, Murphy et al. et al. taught that the adjusting step further comprised copying a datum from a first location to a second location, the first location having a use under the first data storage convention analogous to the use of the second location under the second data storage convention (figs. 7-8, elements 264, 262, 268, 270, 272, 292, 294, 296, 306 (both instances)).

57. As to claim 26, Murphy et al. taught that adjusting step further comprised copying a datum from a third location to a fourth, the third location having a use under the first data storage convention analogous to the use of the third location under the first data storage convention and to the fourth location under the second data storage convention, a program for the copying being programmed to assume that exactly one of the first and third locations is no longer required by the execution of the program (figs. 7-8, elements 264, 262, 268, 270, 272, 292, 294, 296, 306 (both instances)).

58. As to claim 27, Brender et al. taught that the rule for copying data from the first location to the second is determined by examining a descriptor associated with the instruction

before the recognized execution flow or transfer (fig. 5, 310, 312).

59. As to claim 28, Brender et al. taught that the two conventions are two calling conventions (col. 9 line 35 to col. 10 line 40).

60. As to claim 29, Brender et al. taught that one of the two calling conventions is a register-based calling convention (col. 9 lines 36-40, and 67), and the other calling convention is a memory stack-based calling convention (col. 9 line 68).

61. As to claim 30, Goetz et al. taught that the regions are pages (fig. 4, fig. 5, col. 10 lines 4-42) managed by a virtual memory manager (col. 5 lines 1-7).

62. As to claim 31, Goetz et al. taught that the indicator elements are stored in a table (col. 10 lines 4-42) whose entries are indexed by physical page frame number (fig. 4, 50, col. 10 lines 20-22).

63. As to claim 32, Goetz et al. taught that the entry was one entry of a translation look-aside buffer (col. 17 lines 13-16).

64. As to claim 33, Brender et al. taught taking a processor exception in response to the recognition (col. 8 lines 57-67), a handler for the exception programmed to copy a datum from a first location to a second location (fig. 5, col. 8 line 68 to col. 9 line 4, Murphy et al. at figs. 7-8), the first location having a use under the first data storage convention analogous to the use of the second location under the second data storage convention (col. 10 lines 29-40).

65. As to claim 42, Goetz et al. and Brender et al. taught that each indicator element was further designed to store an indication of a calling convention under which the instruction data of the associated region are coded for execution by the processor pipeline (col. 17 lines 24-33, Brender et al. at col. 8 lines 57-65);

and further comprising software programmed to alter the data storage content of a computer using the computer processor to create a program context under the second calling convention that is logically equivalent to a pre-alteration program context under the first calling convention (Brender et al., fig. 5, col. 16 lines 11-45, Murphy et al. figs. 7-8, col. 8 line 56 to col. 10 lines 57);

the memory unit further designed to recognize when program execution has flowed or transferred from a region whose indicator element indicates the first calling convention to a region whose indicator element indicates the second calling convention, and in response to the recognition, to invoke the transition management software (Goetz et al. at col. 17 lines 34-49, Brender et al. at col. 8 lines 57-65).

66. It would have been obvious to combine the teachings of Brender et al. and Murphy et al. with that of Goetz et al. for the same reasons presented, supra. in the rejection of earlier claims under Goetz et al. in view of Brender et al. and Murphy et al.

67. As to claim 43, Brender et al. taught that the memory unit was designed to recognize a single indicator element to indicate both the instruction set architecture and calling convention of a region (Brender et al. at col. 8 lines 60-61 and 66-67, Goetz et al. at col. 17 lines 28-31).

68. As to claim 44, Brender et al. taught that the memory unit and software were designed to effect a transition between instruction boundaries (col. 8 lines 57-65), between execution in a region coded in the first instruction set (col. 8 line 57) using the first calling convention (col. 9 line 67) to execution in a region coded in the second instruction (col. 8 line 59-60) set using the second calling convention (col. 9 line 68), so that code at the source of the flow or transfer may effect the execution transition without being specially coded for code at the destination (col. 10 lines 29-40).

69. As to claim 45, Brender et al. taught that one of the two calling conventions was a register-based calling convention (col. 9 line 67), and the other calling convention is a memory stack-based calling convention (col. 9 lines 68).

70. As to claim 46, Goetz et al. taught that the logical resources for support of the first and second instruction set architectures; are overlaid on the physical resources of the computer processor according to a mapping that assigns corresponding resources of the two architectures to a common physical resource when the resources serve analogous functions in the calling conventions of the two architectures (col. 6 lines 1-25).

71. As to claim 47, Brender et al. taught that a rule for altering the data storage content

from the first calling convention to the second is determined by examining a descriptor associated with the location of execution before the recognized execution flow or transfer (fig. 5, 310, 312).

72. As to claim 49, Brender et al. taught a transition manager (fig. 5) designed to effect a transition between the execution of code coded in instructions of a first instruction set architecture and code coded in instructions of a second instruction set architecture (col. 10 lines 29-40), the transition manager designed to alter a bit representation of a datum from a first representation under the first architecture to a second representation under the second architecture, the alteration of representation being chosen to preserve the meaning of the datum across the change in execution architecture (fig. 5, 316, Murphy et al. in figs. 7-8).

73. As to claim 50, Brender et al. taught circuitry designed to raise an exception when the computer recognizes that execution has flowed or transferred from a region whose indicator element indicates one architecture or execution convention to another (col. 8 lines 57-67).

74. As to claim 56, Brender et al. taught that the instruction data coded for execution by a first of the two instruction set architectures observes a data storage convention associated with the first architecture (col. 2 lines 61-65 and col. 10 lines 50-64), and instruction data coded for execution by a second of the two instruction set architectures observes a second, different, data storage convention associated with the second architecture (col. 10 lines 50-64), a single indicator element indicating both the instruction set architecture and the data storage convention of the associated page (col. 8 lines 60-61 and 66-67, Goetz et al. at col. 17 lines 28-31); and further comprising the step of recognizing when program execution flows or transfers from a page using the first instruction set architecture to a page using the second instruction set architecture (col. 8 lines 57-63), and in response to the recognition, adjusting the data storage content of the computer from the first storage convention to the second (fig. 5, Murphy et al. in figs. 7-8).

75. It would have been obvious to combine the teachings of Brender et al. and Murphy et al. with that of Goetz et al. for the same reasons presented, supra. in the rejection of earlier

claims under Goetz et al. in view of Brender et al. and Murphy et al.

76. As to claim 54, Brender et al. taught that the two conventions were first and second data storage conventions (col. 9 line 35 to col. 10 line 40), and further taught the step of recognizing when program execution has flowed or transferred from a region whose indicator element indicates the first data storage convention to a region whose indicator element indicates the second data storage convention (col. 8 lines 57-63), and in response to the recognition, altering the data storage content of the computer to create a program context under the second data storage convention that is logically equivalent to a pre-alteration program context under the first data storage convention (Brender et al. at fig. 5, col. 16 lines 11-45, Murphy et al. at figs. 7-8, col. 8 line 56 to col. 10 lines 57).

77. As to claim 55, Murphy et al. taught the steps of storing instruction data in a third page, the instruction data of the third page being coded for execution by one of the two architectures, and observing a data storage convention associated with the other of the two architectures; storing 'Indicator elements indicating the data storage convention observed by the instructions of the respective pages; and recognizing each transition of program execution from a page using the first data Storage convention to a page using the second data storage convention, and in response to the recognition, adjusting the data storage content of the computer from the first storage convention to the second, and vice-versa (figs. 7-8, elements 264, 262, 268, 270, 272, 292, 294, 296, 306 (both instances)).

78. As to claim 57, Brender et al. taught that the two conventions were a register-based calling convention (col. 9 line 67) and a memory stack-based calling convention (col. 9 lines 68), and further taught the step of recognizing when program execution has flowed or transferred from a page using the register-based calling convention to a page using the memory stack-based convention (Goetz et al. at col. 17 lines 34-49, Brender et al. at col. 8 lines 57-65), and in response to the recognition, adjusting the data storage content of the computer from the first calling convention to the second (fig. 5, Murphy et al. in figs. 7-8).

79. As to claim 58, Murphy et al. taught that the adjusting step further comprised copying a

datum from a third location to a fourth, the third location having a use under the first data storage convention analogous to the use of the third location under the first data storage convention and to the fourth location under the second data storage convention, a program for the copying being programmed to assume that exactly one of the first and third locations is no longer required by the execution of the program, a bit representation of the datum copied to the second location differing from a bit representation of the datum copied from the first location, the alteration of representation being chosen to preserve the meaning of the datum across the change in data storage convention (figs. 7-8, elements 264, 262, 268, 270, 272, 292, 294, 296, 306 (both instances)).

80. As to claim 59, Brender et al. taught that the rule for copying data from the first location to the second was determined by examining a descriptor associated with the instruction before the recognized execution flow or transfer (fig. 5, 310, 312).

81. As to claim 69, Brender et al. taught that the microprocessor chip was designed to raise an exception when execution flows or transfers from the first region to the second (col. 8 lines 57-67) and further taught exception handler software programmed to handle the exception by explicitly controlling a mode of execution of the instructions (col. 8 line 68 to col. 9 line 4).

82. It would have been obvious to combine the teachings of Brender et al. and Murphy et al. with that of Goetz et al. for the same reasons presented, supra. in the rejection of earlier claims under Goetz et al. in view of Brender et al. and Murphy et al.

83. As to claim 76, Brender et al. taught:

83.1. hardware and/or software designed to (a) to retrieve calling convention indicator elements (fig. 10 and col. 15 lines 6-10) stored in association with respective pages of the memory, each calling convention indicator element indicating which of a register-based calling convention or a memory stack-based calling convention is observed by instructions of the page (col. 8 lines 60-61 and 66-67, Goetz et al. at col. 17 lines 28-31);

83.2. (b) to recognize when instruction execution has flowed or transferred from a page of a

memory-based convention to a page of the register-based calling convention, as indicated by the calling convention indicator elements associated with the respective pages (col. 8 lines 57-63); and,

83.3. (c) in response to the recognition, to alter a storage content of the computer to create a program context under the register-based convention logically equivalent to a pre-alteration program context under the memory-based convention (fig. 5, Murphy et al. in figs. 7-8).

84. It would have been obvious to combine the teachings of Brender et al. and Murphy et al. with that of Goetz et al. for the same reasons presented, supra. in the rejection of earlier claims under Goetz et al. in view of Brender et al. and Murphy et al.

85. As to claim 73, Goetz et al. and Brender et al. taught that each indicator element being further designed to store an indication of a data storage convention under which the instruction data of the associated page are coded for execution by the instruction unit (col. 17 lines 24-33, Brender et al. at col. 8 lines 57-65);

and further comprising software programmed to alter the data storage content of a computer using the microprocessor chip, to create a program context under the second data storage convention that is logically equivalent to a pre-alteration program context under the first data storage convention (Brender et al., fig. 5, col. 16 lines 11-45, Murphy et al. figs. 7-8, col. 8 line 56 to col. 10 lines 57);

the microprocessor chip further designed to recognize when program execution has flowed or transferred from a region whose indicator element indicates the first data storage convention to a region whose indicator element indicates the second data storage convention, and in response to the recognition, to invoke the transition management software (Goetz et al. at col. 17 lines 34-49, Brender et al. at col. 8 lines 57-65).

86. It would have been obvious to combine the teachings of Brender et al. and Murphy et al. with that of Goetz et al. for the same reasons presented, supra. in the rejection of earlier claims under Goetz et al. in view of Brender et al. and Murphy et al.

87. As to claim 74, Murphy et al. taught retrieving instruction data from a third page, the instruction data of the third page being coded for execution by a first of the two architectures, and observing a data storage convention of the second of the two architectures; to retrieve indicator elements indicating the data storage convention observed by the instructions of the respective pages; and to recognize each transition of program execution from a page using the first data storage convention to a page using the second data storage convention, and in response to the recognition, to adjust the data storage content of the computer from the first data storage convention to the second data storage convention, and vice-versa (figs. 7-8, elements 264, 262, 268, 270, 272, 292, 294, 296, 306 (both instances)).

88. As to claim 75, Brender et al. taught recognizing a single indicator element to indicate both the instruction set architecture and calling convention of a page (Brender et al. col. 8 lines 60-61 and 66-67, Goetz et al. at col. 17 lines 28-31).

89. As to claim 77, Brender et al. taught that the two conventions are first and second data storage conventions (col. 9 line 35 to col. 10 line 40); and further taught software programmed to alter the data storage content of a computer using the microprocessor chip, to create a program context under the second data storage convention that is logically equivalent to a pre-alteration program context under the first data storage convention (Brender et al. at fig. 5, col. 16 lines 11-45, Murphy et al. at figs. 7-8, col. 8 line 56 to col. 10 lines 57); the microprocessor chip being further designed to recognize when program execution has flowed or transferred from a region whose indicator element indicates the first data storage convention to a region whose indicator element indicates the second data storage convention, and in response to the recognition, to invoke the transition management software (Goetz et al. at col. 17 lines 34-49, Brender et al. at col. 8 lines 57-65).

90. It would have been obvious to combine the teachings of Brender et al. and Murphy et al. with that of Goetz et al. for the same reasons presented, supra. in the rejection of earlier claims under Goetz et al. in view of Brender et al. and Murphy et al.

91. As to claim 78, Brender et al. taught that the microprocessor chip and software are

designed to effect a transition between instruction boundaries (col. 8 lines 57-65), between execution on page coded in the first instruction (col. 8 line 57) set using the first calling convention (col. 9 line 67) to execution on a page coded in the second instruction set (col. 8 line 59-60) using the second calling convention(col. 9 line 68), the software programmed to effect the execution transition without the code at the source of the transition being specially coded to interface with code at the destination of the transition (col. 10 lines 29-40).

92. As to claim 79, Brender et al. taught that the two conventions are two calling conventions (col. 9 line 35 to col. 10 line 40).

93. As to claim 80, Brender et al. taught one of the two calling conventions is a register-based calling convention (col. 9 lines 36-40, and 67), and the other calling convention is a memory stack-based calling convention (col. 9 line 68).

94. As to claim 81, Goetz et al. taught the physical resources of the microprocessor chip are associated to the logical resources of the first and second calling conventions according to a mapping that assigns corresponding logical resources to a common physical resource when the resources serve analogous functions in the two calling conventions (col. 6 lines 1-25).

95. As to claim 82, Brender et al. and Murphy et al. taught software and/or hardware designed to effect a transition between the execution of code coded under the first calling convention and code coded under the second calling convention, by altering a bit representation of a datum from a first representation under the first calling convention to a second representation under the second calling convention, the alteration of representation being chosen to preserve the meaning of the datum across the change in calling convention (Brender et al. in fig. 5, element 316, Murphy et al. in figs. 7-8).

96. As to claim 83, Murphy et al. taught comprising: software and/or hardware designed to copy a datum from a first location to a second location, the first location having a use under the first calling convention analogous to the use of the second location under the second calling convention (figs. 7-8, elements 264, 262, 268, 270, 272, 292, 294, 296, 306 (both instances)).

97. As to claim 84, Murphy et al. taught software and/or hardware designed to copy a

datum from a third location to a fourth, the third location having a use under the first data storage convention analogous to the use of the third location under the first data storage convention and to the fourth location under the second data storage convention, the software and/or hardware for the copying being programmed to assume that exactly one of the first and third locations is no longer required by the execution of the program (figs. 7-8, elements 264, 262, 268, 270, 272, 292, 294, 296, 306 (both instances)).

98. As to claim 85, Goetz et al. in view of Brender et al. taught hardware and/or software designed to retrieve calling convention indicator elements stored in association with respective pages of the memory (Goetz et al. at col. 14 line 63 to col. 15 line 10), each calling convention indicator element indicating which of a register-based calling convention or a memory stack-based calling convention is observed by instructions of the page (Brender et al. at col. 9 lines 67-68);

to recognize when instruction execution has flowed or transferred from a page using the register-based calling convention to a page using the memory stack-based convention, as indicated by the calling convention indicator elements associated with the respective pages (Goetz et al. at col. 17 lines 34-49), and

in response to the recognition, to alter a storage content of the computer to create a program context under the memory-based convention logically equivalent to a pre-alteration program context under the register-based convention (Brender et al. at fig. 5, Murphy et al. at figs. 7-8).

99. It would have been obvious to combine the teachings of Brender et al. and Murphy et al. with that of Goetz et al. for the same reasons presented, supra. in the rejection of earlier claims under Goetz et al. in view of Brender et al. and Murphy et al.

100. As to claims 87-93, 96-103, 110-112, 116-117, 118-126, and 133, they do not teach or define above the invention claimed in the previously rejected respective claims and are therefore rejected under Goetz et al. in view of Brender et al. and Murphy et al. for the same reasons set fourth in the previous claim rejections, supra.

101. It would have been obvious to combine the teachings of Brender et al. and Murphy et al. with that of Goetz et al. for the same reasons presented, supra. in the rejection of earlier claims under Goetz et al. in view of Brender et al. and Murphy et al.

102. As to claim 88, Murphy et al. taught that the data movement includes at least one of. copying data from a general register to a memory stack (fig. 7, 264, 262, 268 270, 272); and copying data from a memory stack to a general register (fig. 8, 292, 294, 296, 306 (both instances)).

103. Applicant's arguments with respect to claims 1-133 have been considered but are deemed to be moot in view of the new grounds of rejection.

104. Claims 20, 34-36, 48, 60, 67, 86, and 127 are objected to as being dependent upon a rejected base claim, but would render the base claim allowable if bodily incorporated into the base claim such that the new base claim includes all of the original limitations of the base claim, any intervening claims, and the objected claim.

105. The prior art made of record and not relied upon is considered pertinent to Applicant's disclosure.

106. A shortened statutory period for response to this action is set to expire 3 (three) months and 0 (zero) days from the mail date of this letter. Failure to respond within the period for response will result in **ABANDONMENT** of the application (see 35 USC 133, MPEP 710.02, 710.02(b)).

107. Any inquiry concerning this communication or earlier communications from the Examiner should be directed to Richard Ellis whose telephone number is (703) 305-9690. The Examiner can normally be reached on Monday through Thursday from 7am to 5pm.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Eddie Chan, can be reached on (703) 305-9712. The fax phone number for the USPTO is: (703)872-9306.

Any inquiry of a general nature or relating to the status of this application should be directed to the Group receptionist whose telephone number is (703) 305-3900.

Richard Ellis
February 9, 2004

RICHARD L. ELLIS
PRIMARY EXAMINER